Java Native Integration

Reaching out to other Apple APIs

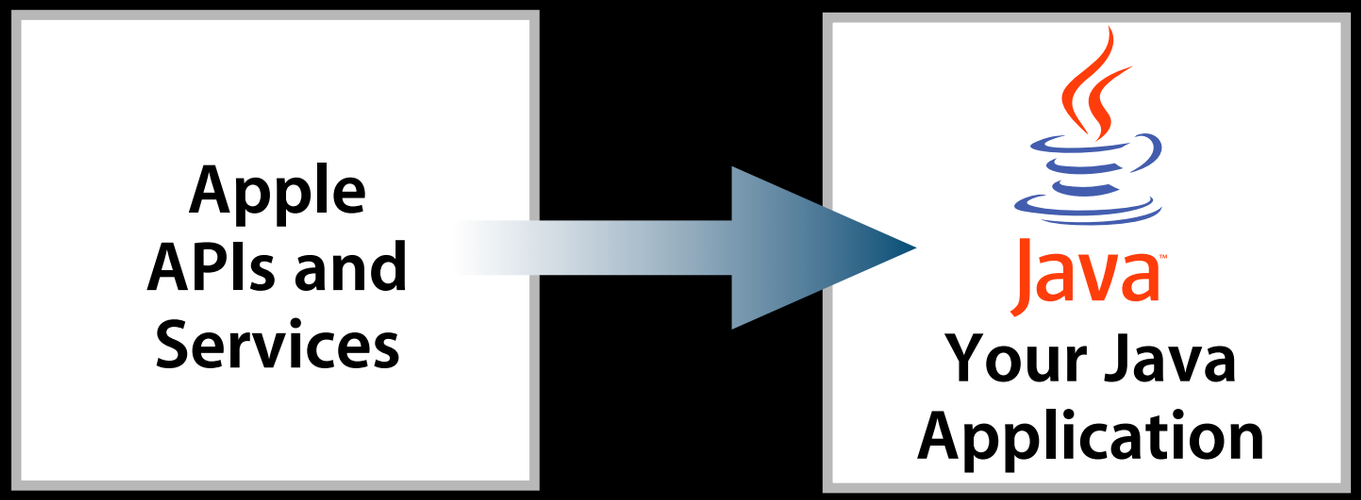
Session 620 i.e.

Java Native Integration

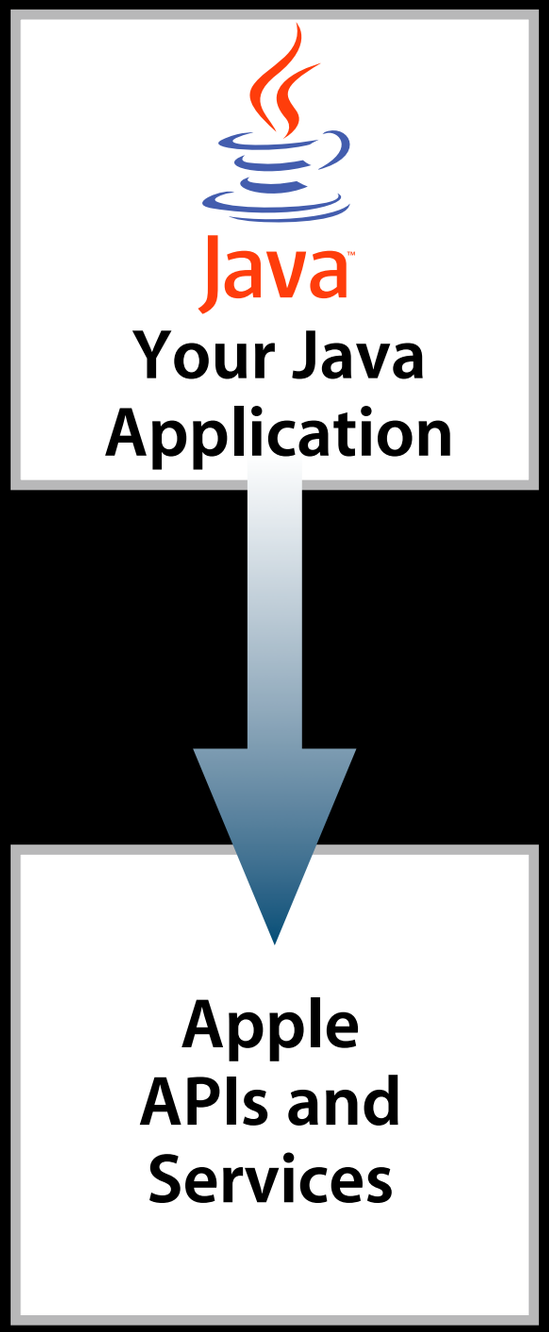
Reaching out to other Apple APIs

Blaine Garst
Manager, Java Runtimes
blaine@apple.com

Let’s start with your Application



Apple services reach
in and benefit your app...



Your app can reach out and use Apple APIs and Services

Overview

- Demo: Things you get for free
  - AppleScript GUI, Inkwell
- Threading architecture
- Examples
  - Java launchers
  - Embedded NSMovieView
  - Disc Recording
  - Address Book
- Demo
- Q/A

Things you Get for Free

- Cocoa Input Methods
  - Multi-language input
  - Inkwell
- Accessibility implementation in Cocoa peer classes
  - Accessibility!
  - AppleScript GUI scripting
- Menu items
  - Services menu
    - Spelling
    - Speech

Demo

Greg Bolsinga

Embedding Applets in Cocoa

- How do I add a Java applet to my Cocoa application?
- Use the WebKit SDK!
- Just add a WebView to the application and have it load an applet's web page

Want More Information? Go To:

420 Internet Technologies: Advanced WebKit APIs

Extended Apple Scripting

- Standard Scripting
  - openApp, openDoc, printDoc, reopenApp, quitApp
  - Automatic GUI scripting of accessibility items
- Advanced Scripting
  - Dictionary of nouns and properties and verbs
  - Cocoa offers wrappers for Apple Events
    - NSScript{...} deals with Open/Close, documents
    - Key/Value coding deals with “properties”
  - Only available to Cocoa Java programmers

Threading Architecture

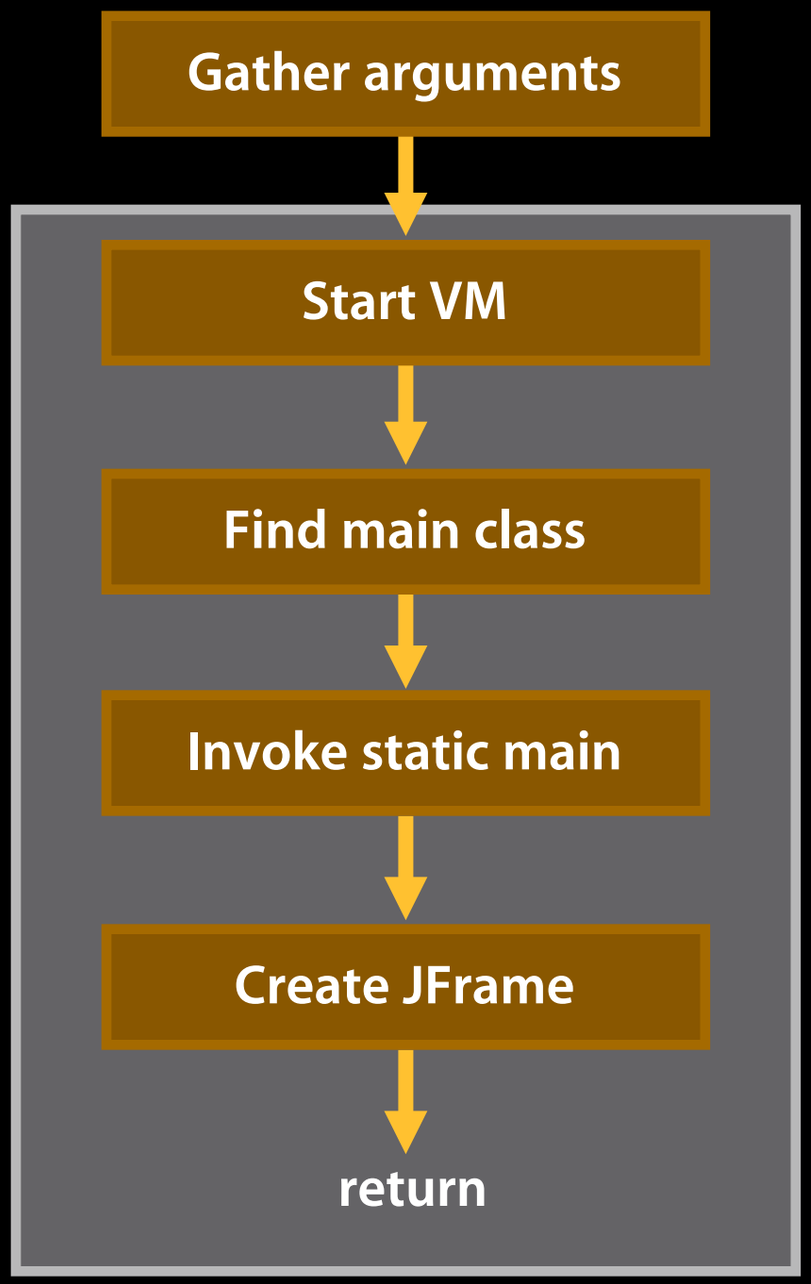
What's a Thread?

```
mach_msg_trap ()
RunCurrentEventLoopInMode ()
_DPSNextEvent ()
-[NSApplication run]
-[StartupUtility run]
__CFRunLoopRun()
main()
```

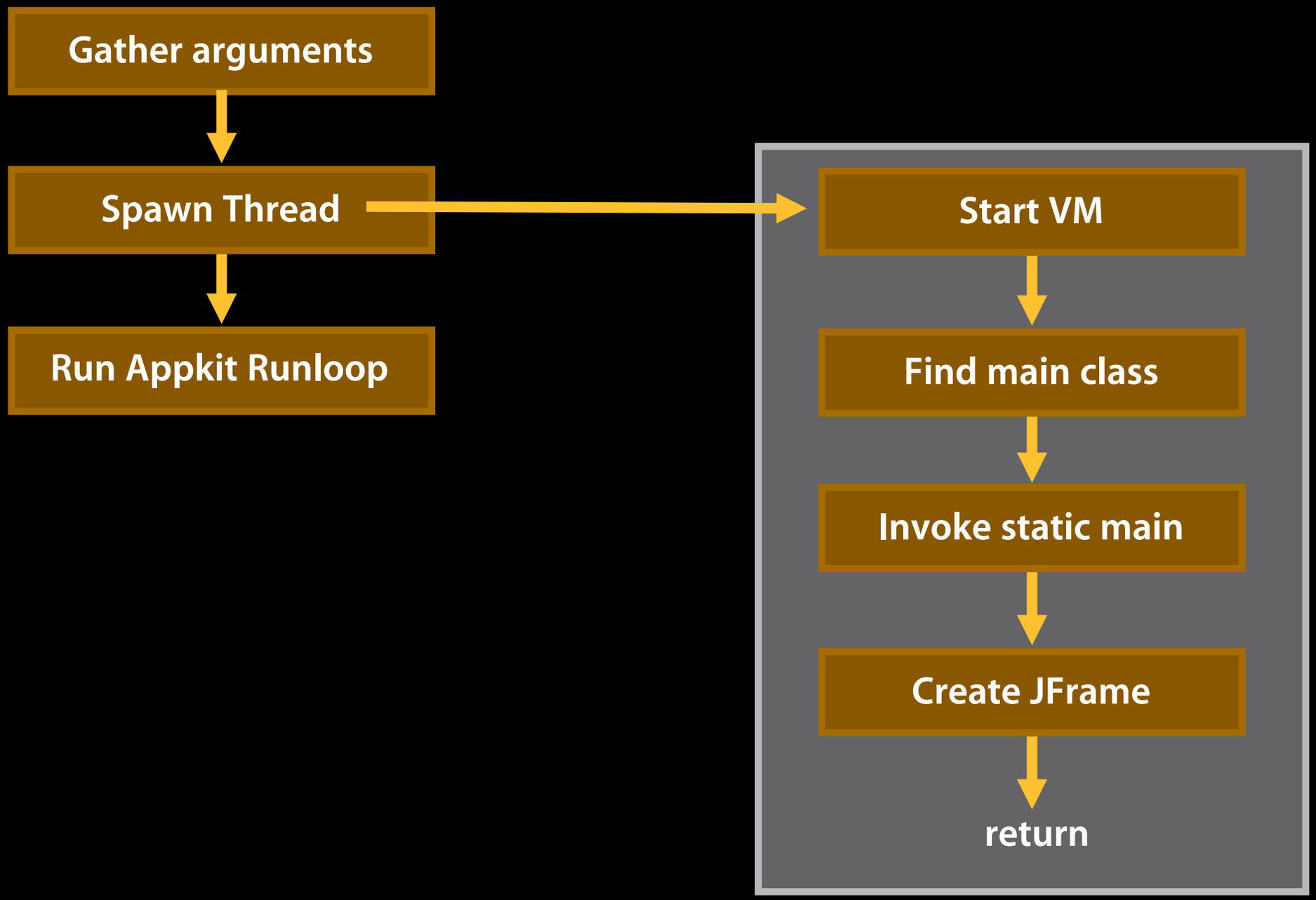
- Kernel: registers + stack
- Processes start with one
- First "spawns" others
- Several in each process

Library initializers force duties upon first thread!

Java Launch Sequence—Standard



Java Launch Sequence—Apple



Apple Java Threading Architecture

 # 1
AppKit (main)

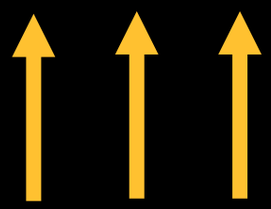
#2
Java main

#3
AWT Event Queue Thread

#4...
Other threads

mach_msg_trap ()
RunCurrentEventLoopInMode ()
_DPSNextEvent ()
-[NSApplication run]
-[StartupUtility run]
__CFRunLoopRun()
main()

myapp.doinput()
myapp.main()
JNI_InvokeStaticVoid()



Keyboard, mouse, ...

Launchers—Getting Java Started

- Standard mechanisms
  - /usr/bin/java
  - App launcher stub: Contents/MacOS/MyApp
    - Uses JVMVersion to pick 1.3.1 vs. 1.4.1
- Both contain JNI code to start a Virtual Machine
  - JNI routines found in JavaVM.framework

Why Write a Launcher?

- Try not to
- However, if you have
  - Existing threads of legacy code
  - Properties that change on a per-launch basis
  - Need a non-standard main

    ...
- Start with our example code
  - developer.apple.com/java

Embedding NSMovieView

Embedding Cocoa Components

- AWT peers built upon Cocoa's AppKit
- com.apple.eawt.CocoaComponent
  - Fields AppKit events
  - Paints on AppKit (main) thread
  - Subclasses implement createNSView
- Has architecture for thread communication
- NSMovieView developer example code

CocoaComponent: Generic “Peer”

Java Peer:

class JMovieView extends CocoaComponent

Cocoa View:

@interface CocoaMovieView : NSMovieView

JMovieView (Java)

```
public class JMovieView extends CocoaComponent {


    static {
        try {
            System.loadLibrary("sample");
        }
        catch(Throwable t) {
            t.printStackTrace();
        }
    }

    public native int createNSView();
```

JMovieView (Java)

```
public static final int kSetMovieUrl    = 0;
public static final int kPlayMovie      = 1;
public static final int kStopMovie      = 2;

public void setMovieUrl(String url) {
    sendMessage(kSetMovieUrl, url);
}

public void play() {
    sendMessage(kPlayMovie, null);
}

public void stop() {
    sendMessage(kStopMovie, null);
}
```

CocoaMovieView (Objective-C)

```
@implementation CocoaMovieView : NSMovieView { }

// CocoaComponent.sendMessage comes here
-(void)awtMessage:(jint)messageID message:(jobject)message env:(JNIEnv *)env
{
    switch (messageID) {
        case NSMovieView_kSetMovieUrl:
            [self setMovie:[[NSMovie alloc] initWithURL:[NSURL
URLWithString:convertToNSString(env, message);] byReference:YES]];
            break;
        case NSMovieView_kPlayMovie:
            [self start:self];
            break;
        case NSMovieView_kStopMovie:
            [self stop:self];
            break;


    }
}

@end
```

CocoaMovieView (Objective-C)

```
JNIEXPORT jint JNICALL Java_NSMovieView_createNSView
  (JNIEnv *env, jobject nsMovieView)
{
    movieView = [[CocoaMovieView alloc] init];

    [movieView showController:TRUE adjustingSize:NO];

    // Return a pointer to the custom NSView (the view
must have a retain count of at least 1)
    return (jint) movieView;
}
```

Disc Recording

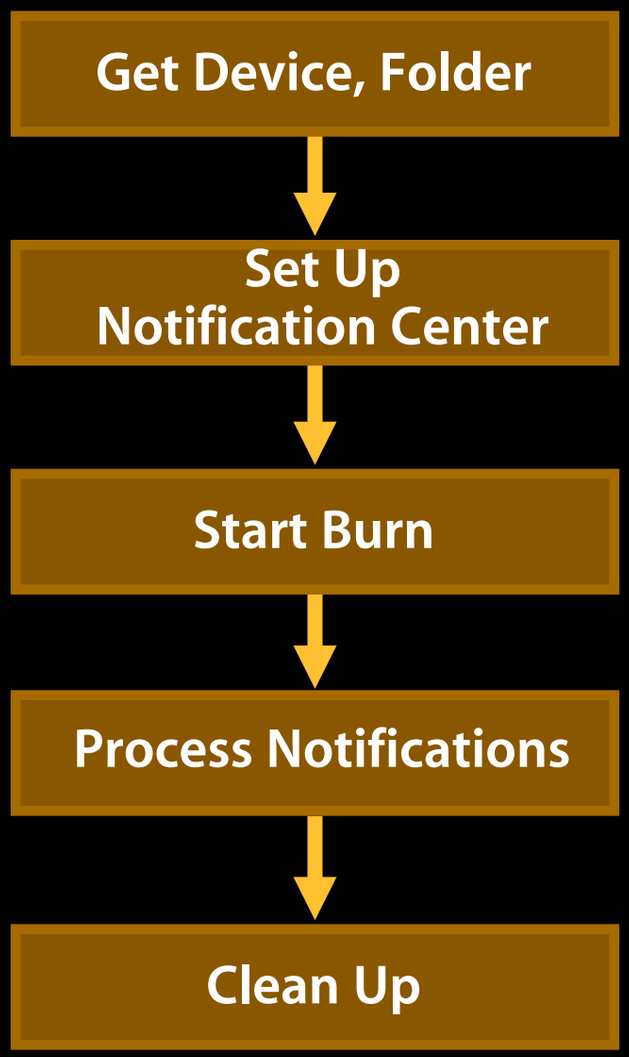
Disc Recording

- C-based DiscRecording.framework
- Objective-C based DiscRecordingUI.framework

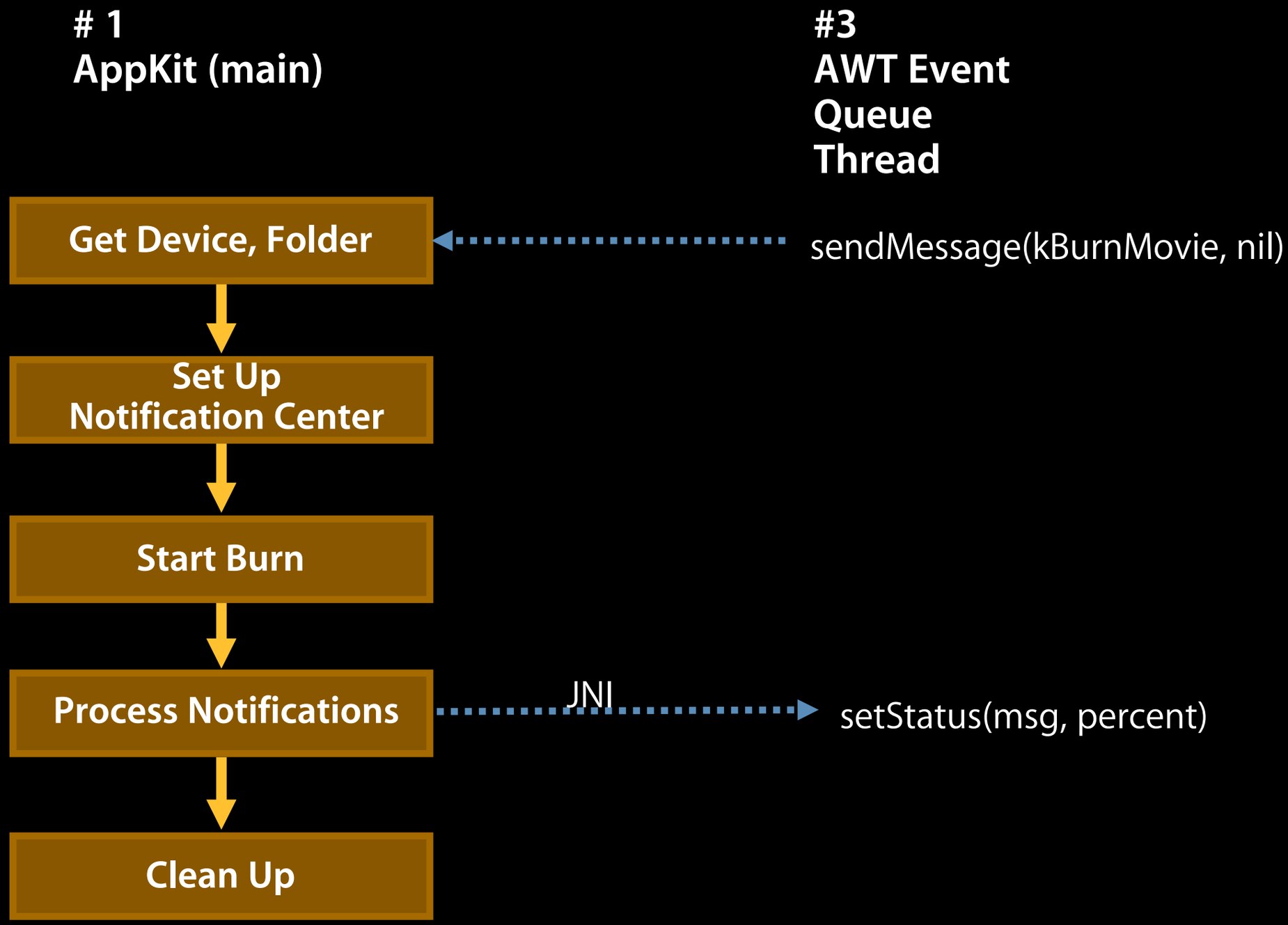
- Both use a runloop to drive work and notifications
  - *GUI work is very tricky—should be avoided!*

- How hard would it be to add a Burn button?

Basic Burn Logic (From Example)



Java AWT Example



Demo

Blaine Garst
blaine@apple.com

Address Book

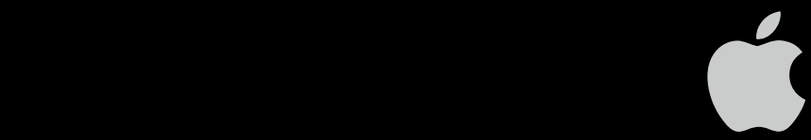
Address Book Example

```
JNIEXPORT jstring JNICALL Java_JMovieView_getMeFirstName
  (JNIEnv *env, jclass jmovieviewClass) {

    ABPerson *me =
        [[ABAddressBook sharedAddressBook] me];

    NSString *firstName =
        [me valueForProperty:kABFirstNameProperty];

    return makeJString(env, firstName);
}
```



Wrap Up

Roadmap

420 Internet Technologies: Advanced WebKit APIs

The Mission
Friday, 9:00am

FF014 Java Feedback Session

How are we doing?

The Marina
Friday, 10:30am

625 Maximizing Java Performance on Mac OS X

What we've done, what you can do

Russian Hill
Friday, 3:30pm

Q&A

Alan Samuel
Java Technologies Evangelist
bluker1@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference